第 2 章
Python 语言财务数据分析

学习目标

1. 了解安装 Python 程序的基本步骤，能够配置 Python 环境并运行 Python 代码。

2. 了解 Python 语言基础语法、变量的定义与使用，能编写符合 Python 语法规范的代码。

3. 熟悉 Python 语言中运算符的使用方法，掌握 Python 语言中各种数据类型的创建、修改等操作，理解条件判断语句 if、多分支选择结构，掌握 for 循环、while 循环，了解 Python 语言常用函数的定义方法。

4. 了解财务数据的获取方式，掌握数据清洗与处理的操作方法。

思政目标

1. 树立正确的价值观，将数据分析方法的应用与中国特色社会主义建设的理论和实践相结合，要科学合理地使用数据分析方法解决实际问题。

2. 遵守《中华人民共和国网络安全法》《中华人民共和国数据安全法》《中华人民共和国个人信息保护法》等相关法律法规，遵守职业道德，将数据分析方法与实事求是的理念相结合。

3. 牢记数据分析服务于社会的使命，加强学习，能创造更多、更便捷的数据分析方法，服务于科学研究，服务于社会。

2.1 导言

Python 作为一门简单易学的语言，其基础语法、数据结构、程序流程控制和函数都非常直观易懂。数据分析是 Python 语言的强项，现已广泛应用于财务领域。为帮助学生牢固地掌握 Python 语言基础知识，为后续的财务数据分析奠定扎实的基础，本章全面介绍 Python 语言数据分析方法，内容涉及 Python 语言基础语法，数据获取、数据读取、数据筛选、数据清理等。通过详细讲解知识点和分析案例，学生能系统掌握 Python 语言数据

分析功能，为后续的数据探索奠定基础。

鉴于现代财务领域的数字化转型和传统教学工具的局限性，当前金融行业及企业财务部门已广泛采用 Python 语言进行自动化处理、海量数据分析和复杂建模，而传统《财务管理》《投资学》等教材主要依赖理论和 Excel 工具，存在显著的技能断层。Excel 在处理大规模数据集、构建高级模型及实现流程自动化等方面力不从心，亟须引入 Python 语言这一强大工具来突破瓶颈。Python 语言凭语言借其丰富的数据处理（Pandas）、统计分析（SciPy）和机器学习（Scikit-learn）库，使学生能真正实践从多源获取数据到清洗、分析、建模的全流程。

本章节旨在提升学生的分析效率与深度，培养其处理更大规模、更高维度数据的能力，揭示传统方法难以发现的模式和风险，深化对财务核心理论的理解与应用能力。同时为学习进阶课程及进行学术研究奠定坚实的技术基础，是连接经典财务理论与数字化实践、培养适应未来挑战的复合型人才不可或缺的关键学习环节。

2.2 Python 语言简介

Python 是一种计算机程序设计语言，因其功能强大、语法结构简单，对于初学者来说通俗易懂、易学易用，是目前全球最热门的编程语言之一。吉多·范·罗苏姆（Guido van Rossum）于 20 世纪 80 年代发明了 Python 语言。Python 是一种面向对象、解释型的程序设计语言，用途十分广泛，既可以方便有效地实现面向对象的系统、网络编程，也可以用来构建机器学习模型。数据分析及可视化是 Python 语言的强项，现已广泛应用于科学计算、数据分析等诸多领域。

2.2.1 Python 语言的特点

1. Python 语言语法简单

在 Python 语言的 Lib 目录中隐藏了一首诗《Python 之禅》，这首诗表达了 Python 语言的思想：简单、明确、优美。Python 与其他的计算机语言（C 语言、Java、VB、C++）相比，语法更为简洁、简单，如用 Python 语言编写的输出“hello”程序只需 1 行代码，而用 C 语言编写此程序则需 5～7 行代码。

2. Python 是自由开源的编程语言

Python 是开源免费的编程语言，用户可以自由地发布其副本，阅读源代码并可对其进行修改应用于新的自由软件中。Python 语言还可轻松地与其他语言（特别是 C 语言、C++ 语言）连接在一起，完成各种工作。

3. Python 语言拥有丰富而强大的第三方库

Python 语言提供了大量的第三方扩展库，按其用途大致可分为以下几类：数据分析类、爬虫类、Web 开发类、机器学习类和数据库操作类。以数据分析类为例，Python 语言有强大的科学计算“三剑客”：NumPy、Pandas 和 Matplotlib，这些库不仅可以对数据进行

收集、处理、可视化，其自带的分析方法模型使数据分析变得简单高效。同样作为数据分析工具的 SQL、Excel，虽能实现基本的数据分析工作，但在数据量较大的情况下，处理会很麻烦而且处理速度也会变慢。

2.2.2　Python 语言常用的第三方库

1. NumPy

NumPy 是 Python 语言科学计算的基础库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库，包括统计学、线性代数、矩阵数学等，很多 Python 数据计算工作库都依赖于它。

2. SciPy

SciPy 库是基于 Python 语言的科学计算工具包，提供了丰富的数学、科学和工程计算功能。它主要包含统计、优化、线性代数、傅里叶变换、信号和图像处理、常微分方程求解等功能。SciPy 库依赖于 NumPy 库，NumPy 库提供了方便和快速的 N 维数组操作。它们可以一起运行在所有流行的操作系统上，安装简单，使用免费。

3. Matplotlib

Matplotlib 是 Python 语言的一个 2D 绘图库，能够创建静态、动态和交互式图表，广泛用于数据可视化。

4. Pandas

Pandas 库提供了灵活的数据结构，如进行结构化数据分析的二维表格型数据结构 DataFrame，能提供类似于数据库中的切片、切块、聚合、选择子集等精细化操作，使数据清洗和分析变得简单高效。

2.2.3　Python 程序安装

学习 Python 语言，首先要安装 Python 程序。这里，不推荐直接安装 Python 程序，建议通过安装 Anaconda 来安装 Python 程序。Anaconda 集成了很多关于 Python 语言的第三方库，能避免以后的库安装问题。接下来，介绍两种比较简单的 Anaconda 下载安装方法，按步骤操作即可。

第一种方法，在浏览器中打开 Anaconda 的官方下载网站（https：//www.anaconda.com），如图 2−1 所示，单击右上角的 Free Download 按钮。然后出现登录邮箱或跳过注册页面（如图 2−2 所示），在页面右侧，可选择登录邮箱后下载，也可跳过注册登录邮箱环节直接下载 Anaconda。登录邮箱下载，需在 Email Address 文本框中输入邮箱，并勾选同意复选框，单击 Submit 按钮，跳转至下载页面；跳过注册登录邮箱环节下载，单击 Skip registration 按钮，进入下载页面，如图 2−3 所示。最后，根据计算机系统配置选择相应的版本下载，这种方法下载速度较慢。

图 2-1　Anaconda 的官方下载网站

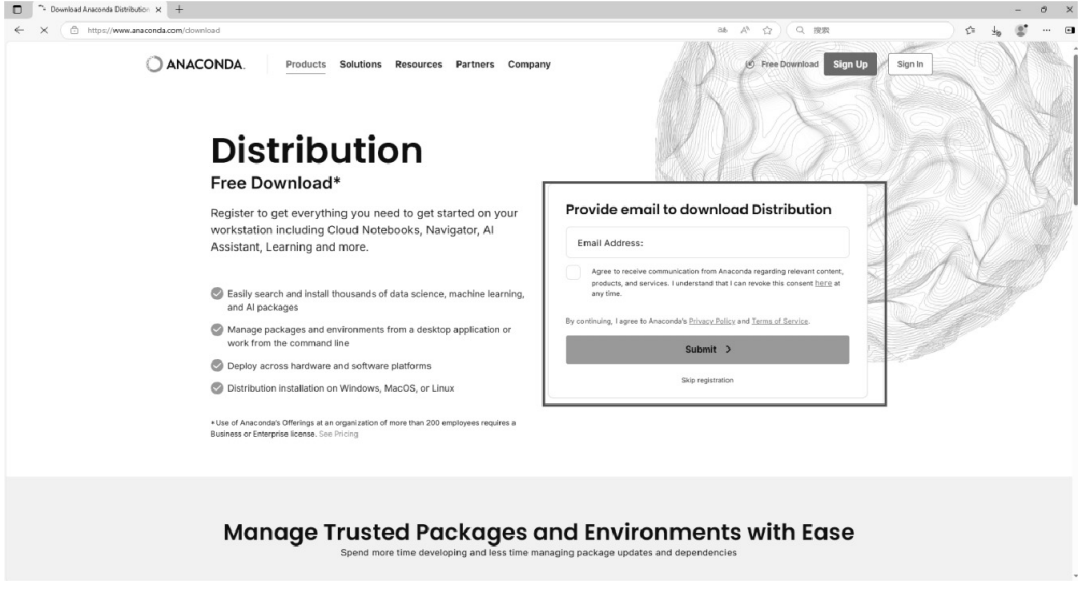

图 2-2　Anaconda 的登录邮箱或跳过注册页面

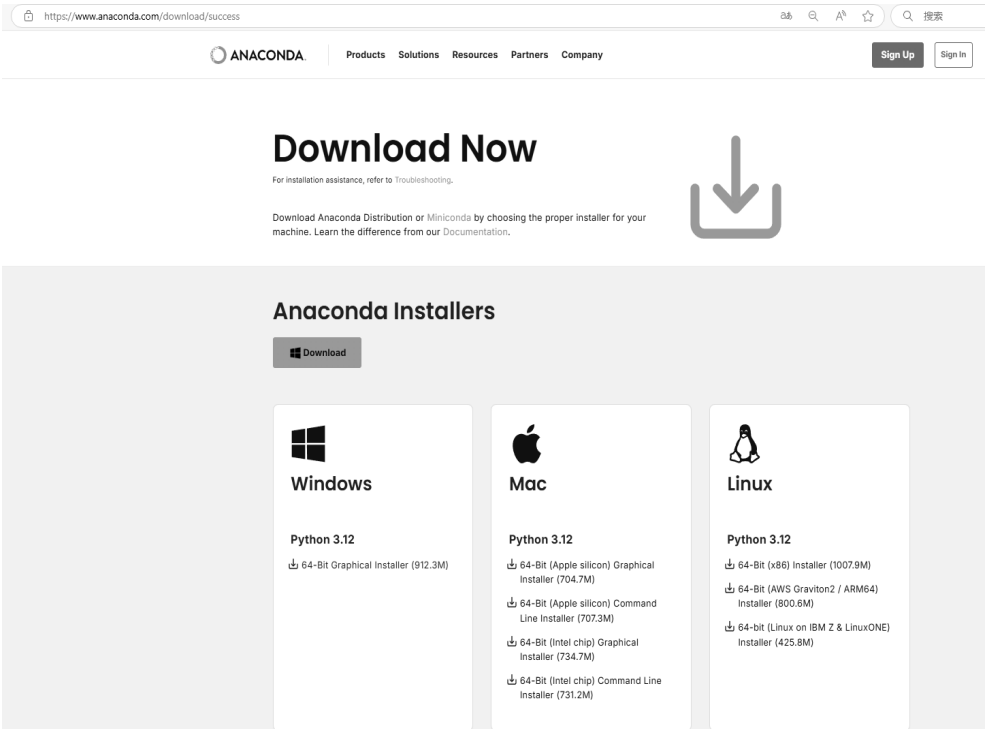

图 2-3　Anaconda 官方网站的下载页面

第二种方法，打开清华大学开源软件镜像站网址 https：//mirrors.tuna.tsinghua.edu.cn/，如图 2-4 所示。选择 anaconda 选项，将出现不同版本的 Anaconda，可选择合适的版本下载安装，如图 2-5 所示。

图 2-4　清华大学开源软件镜像站

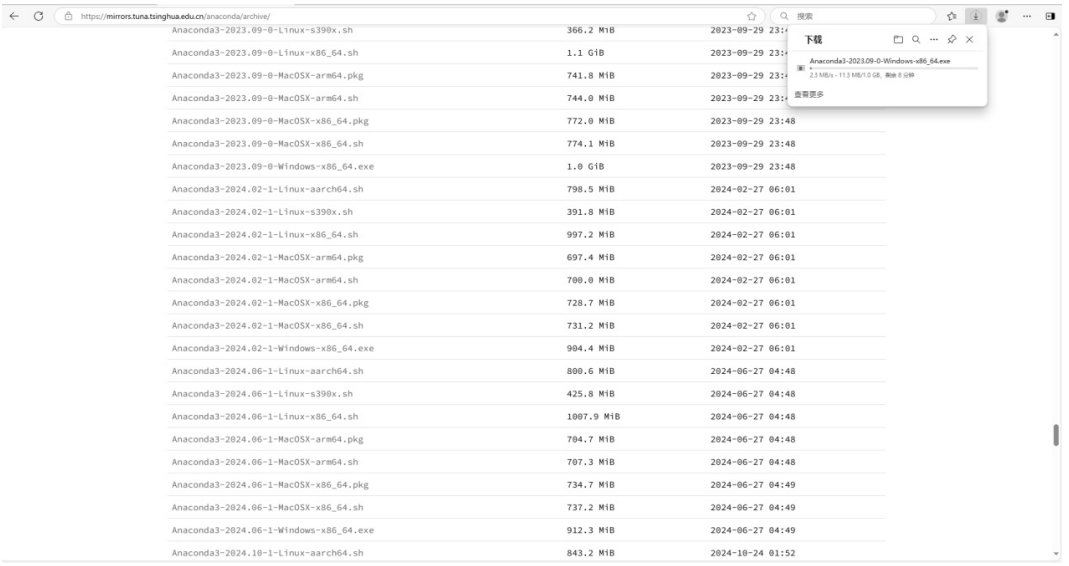

图 2-5　清华大学开源软件镜像站的 Anaconda 下载页面

通过上述方法下载 Anaconda 后，双击已下载的 Anaconda 程序，按指引完成安装。当出现如图 2-6 所示的对话框时，设置安装的 Anaconda 程序可以为当前计算机上的所有用户使用，然后，出现图 2-7 所示对话框，可根据安装需要选择勾选三个复选框，三个复选框分别是指：是否创建开始菜单快捷方式，是否将 Anaconda 注册为 Python 语言推荐系统，安装完成后是否清除包缓存。

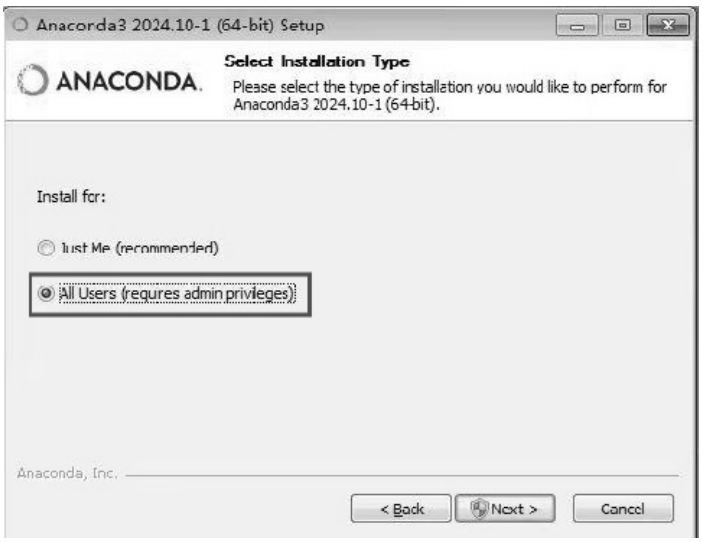

图 2-6　Anaconda 的安装类型

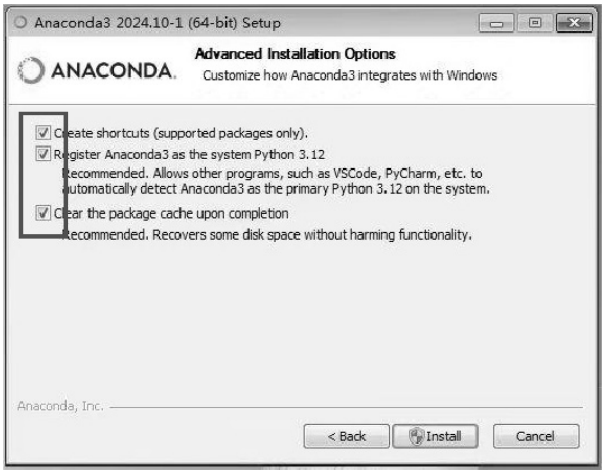

图 2-7　Anaconda 安装的选项设置

2.3　Python 语言基础知识

2.3.1　基础语法

基础语法在编写 Python 代码时，所有输入的代码必须在英文半角模式下输入，不能使用英文全角或中文全角格式。以下介绍一些 Python 基础语法要点，涵盖注释、变量、关键字、缩进、运算符等核心概念。

1. 注释

注释又称批注，Python 语言中的注释是用于解释和说明代码的功能、逻辑和意图的文本。它不会被 Python 解释器执行，初学者可以通过注释理解代码的意图。注释类型分为单行注释和多行注释两种。

1）单行注释

以#符号开头，后面跟随注释内容，它适用于对单个语句或代码块的简短说明（#和注释内容以一个空格隔开）。

［例 2-1］

```
print("Hello World")  # 输出"Hello World"
```

运行结果如图 2-8 所示。

图 2-8　[例 2-1]运行结果

第 2 章
例题 Python 代码

2）多行注释

以一对三重引号（'''或"""）来解释说明 Python 文件、类或方法。

［例 2-2］

```
"""
输出 1
输出 2
"""
print(1)
print(2)
```

运行结果如图 2-9 所示。

图 2-9 ［例 2-2］运行结果

2. 变量

变量来源于数学，是计算机语言中用于存储数据的容器。如一次函数 $Y=X+1$ 中的 X、Y 都是变量，X 为自变量，Y 为因变量。Python 语言中变量赋值类似于函数的假设，赋值的定义格式很简单，可单个变量赋值、多个变量赋值，也可多个变量赋相同的值，格式如下：

```
变量名=值
变量名 1，变量名 2，变量名 3=值 1，值 2，值 3
变量名 1=变量名 2=变量名 3=值
```

上式中，=代表赋值，表示将等号右边的值赋予左边的变量。变量名必须遵循以下规则：变量名只能包含字母、数字和下划线；变量名不能以数字开头；变量名需区分英文字母的大小写；不能使用 Python 语言的关键字作为变量名，如 if、for、class 等；变量名的长度没有限制。

［例 2-3］

```
X=10  # 定义变量 X，值为 10
print(X)  # 输出 X
Y=X+5  # 定义变量 Y，值为 X+5
print(Y)  # 输出 Y
```

运行结果如图 2-10 所示。

图 2-10 ［例 2-3］运行结果

3. 关键字

关键字是 Python 语言中具有特殊意义的保留字，它们不能被用作常量、变量或其他任何对象的名称。每一种程序语言都有保留字，是被编程语言内部定义并保留使用的标识符。使用关键字可以调用 Python 语言提供的内置功能和语法结构。熟记 Python 语言的 33 个关键字，有助于编写符合语法标准的程序，避免出现语法错误。关键字见表 2-1。

表 2-1　关键字

and	def	for	is	raise	True
as	del	from	lambda	return	False
assert	elif	global	nonlocal	try	None
break	else	if	not	while	class
except	import	or	with	continue	finally
in	pass	yield			

4. 缩进

缩进是指代码行开头处的空格，Python 语言使用缩进来定义代码块的层次结构，有助于学生更好地理解代码。如函数定义、条件语句、循环语句，若未采用合理的代码缩进，系统编译时将抛出 IndentationError 异常，并提出代码修改建议。缩进的空格数量是可变的，约定俗成的规则是每一级缩进使用 4 个空格。正确的缩进是确保 Python 解释器能够正确理解代码结构的基础。

［例 2-4］

```
X=10  # 定义变量 X，值为 10
if X>0:  # 如果 X 大于 0
    print('正数')  # 输出'正数'
else:  # 否则
    print('负数')  # 输出'负数'
```

运行结果如图 2-11 所示。

图 2-11 ［例 2-4］运行结果

5. 运算符

1）算术运算符

算术运算符是处理四则运算的符号，如加、减、乘、除。Python 语言基本算术运算符见表 2-2。

表 2-2　Python 语言的基本算术运算符

运算符	描述
+	加——两个对象相加
−	减——得到负数或是一个数减去另一个数
*	乘——两个数相乘或是返回一个被重复若干次的字符串
/	除——x 除以 y
%	取余——返回除法的余数
**	幂——返回 x 的 y 次幂
//	取整除——返回商的整数部分

［例 2-5］

```
X="name"+"A"  #定义变量 X，值为"name"+"A"
print(X)  #输出'X'
```

运行结果如图 2-12 所示。

```
nameA
```

图 2-12 ［例 2-5］运行结果

［例 2-6］

```
X="name!"*3 #定义变量 X，值为"name!"*3
print(X)  #输出'X'
```

运行结果如图 2-13 所示。

```
name!name!name!
```

图 2-13 【例 2-6】运行结果

2）赋值运算符

赋值运算符主要用于为变量等赋值，可以直接把赋值运算符右边的值赋值给左边的变量，也可以进行某种运算后再赋值给左边的变量。熟练使用赋值运算符，可以提高代码效率。Python 语言常用的赋值运算符见表 2-3。

表 2-3 Python 语言常用的赋值运算符

运算符	描述	实例
=	赋值运算符	a=5 将 5 赋值给 a
+=	加法赋值运算符	a+=5 等效于 a=a+5
-=	减法赋值运算符	a-=5 等效于 a=a-5
*=	乘法赋值运算符	a *=5 等效于 a=a*5
/=	除法赋值运算符	a/=5 等效于 a=a/5
%=	取模赋值运算符	a%=5 等效于 a=a%5
**=	幂赋值运算符	a **=5 等效于 a=a**5
//=	取整除赋值运算符	a//=5 等效于 a=a//5

3）比较运算符

比较运算符又称关系运算符，用于对变量或者表达式的结果进行大小、真假等比较，如果比较结果为真返回 1，表示 True；反之返回 0，表示 False。比较运算符可以进行数值比较和条件判断。Python 语言常用的比较运算符见表 2-4。

表 2-4　Python 语言常用的比较运算符

运算符	描述	实例
==	等于——比较对象是否相等	a=5，b=6，a==b 返回 False
！=	不等于——比较两个对象是否不相等	（a！=b）返回 True
＞	大于——返回 x 是否大于 y	（a＞b）返回 False
＜	小于——返回 x 是否小于 y	（a＜b）返回 True
＞=	大于等于——返回 x 是否大于等于 y	（a＞=b）返回 False
＜=	小于等于——返回 x 是否小于等于 y	（a＜=b）返回 True

4）逻辑运算符

数学中的逻辑运算和 Python 语言中的逻辑运算有些类似。Python 语言的逻辑运算符实现逻辑与、或、非运算，常用于构建条件表达式和复杂逻辑（见表 2-5）。

表 2-5　Python 语言的逻辑运算符

运算符	逻辑表达式	实例
and	a and b	a 和 b 同为真时结果为 True，否则为 False
or	a or b	a 和 b 都为假时结果为 False，否则为 True
not	not a	如果 a 为 True，结果为 False；如果 a 为 False，结果为 True

5）成员运算符

对于具有集合概念的对象如数字序列、字符串、列表、元组、字典，可以通过成员运算符进行快速判断，Python 语言的成员运算符见表 2-6。

表 2-6　Python 语言的成员运算符

运算符	描述
in	如果在指定的序列中找到值，则返回 True，否则返回 False
not in	如果在指定的序列中没有找到值，则返回 True，否则返回 False

2.3.2　数据类型

Python 语言常用的 6 种数据类型包括数字（number）、字符串（string）、列表（list）、元组（tuple）、字典（dictionary）和集合（set）（见表 2-7）。这些数据类型在 Python 语言中各有其用途和特性，了解它们的使用方法和限制条件，可以更好地进行编程。

表 2-7　Python 语言的 6 种数据类型

类型	描述	实例
数字	整数（int）、浮点数（float）、复数（complex）、布尔值（bool）	不同数字类型之间可以使用函数相互转换

续表

类型	描述	实例
字符串	描述文本的一种数据类型	由任意数量的字符组成，基本格式为 'hello'
列表	有序的可变序列	列表没有固定大小，基本形式为［1，2，3，4］
元组	有序的不可变序列	元组是不可变的，基本形式为（1，5，7，10）
字典	无序 Key-Value 集合	同一字典内键（Key）必须是互不相同的，基本形式为{'固定资产'：250 000，'无形资产'：80 000}
集合	无序的不重复集合	基本形式为{'固定资产'，'无形资产'，'存货'}

1. 数字

Python 语言中的数字类型包括：整数、浮点数、复数和布尔值。

1）整数

整数可以是正整数、0 或负整数。在 Python 语言中，整数大小没有限制，可以是一个无限大的整数，仅受限于内存。除了常见的十进制外，整数类型还可以用二进制、八进制和十六进制表示，且各进制的数字之间还可以相互转换。

［例 2-7］

```
X=11*100  #定义变量 X，值为 11*100
print(X)  #输出'X'
```

运行结果如图 2-14 所示。

图 2-14 ［例 2-7］运行结果

2）浮点数

浮点数表示带有小数点的数字，并且小数点后的数字可以是 0。由于计算机内存中存储浮点数的位数有限，所以超过一定长度后，末尾将采取近似值处理，常在第 16 位作近似处理。因此，浮点数不一定是精确值。

3）复数

复数是一种数学上的数，可以表示为 $z=a+b$i（a 和 b 都是实数），其中 a 称为实部，b 称为虚部，i 称为虚数单位。在 Python 语言中，complex（ ）函数用于创建一个值为 real+imag * j 的复数，其中 real 为实部，imag 为虚部。

［例 2-8］

```
complex_num=5 + 6j  # 创建一个复数
print(complex_num.real)  # 输出复数的实部
print(complex_num.imag)  # 输出复数的虚部
```

运行结果如图 2-15 所示。

图 2-15 ［例 2-8］运行结果

4）布尔值

布尔值只有 True 和 False 两种。布尔又称逻辑，在 Python 语言中用 True 和 False 表示逻辑判断。该数据是一种特殊的整数类，True 可以用 1 替换，代表真；False 可以用 0 替换，代表假。

2. 字符串

字符串由任意字符组成，用于表示文本数据，字符串的内容可以包含字母、标点、特殊符号、中文等全世界所有的文字字符。在 Python 语言中，可以用单引导（'）、双引号（"）或三引号（'''）来定义。字符串是不可变的，即一旦创建，其内容不能直接修改。

字符串和数字的核心区别是需要知道 1 和"1"是不同的数据类型，1 是数字中的整数，数字间可进行加减乘除运算；"1"加了引号表示的是字符串，也就是常说的文本内容。字符串的一个特点就是它的两边有单引号或双引号。

不同的数据类型是不能进行相互运算的。

［例 2-9］

```
X='1'+1  # 定义变量 X，值为'1'+1
print(X)  #输出'X'
```

运行结果如图 2-16 所示。

```
TypeError: can only concatenate str (not "int") to str
```

图 2-16 ［例 2-9］运行结果

3. 列表

列表是一种可变的数据结构，可以包含不同类型的元素，列表中的元素可以是数字，可以是字符串，还可以是列表等。列表通过方括号表示，元素之间用逗号隔开。列表的基本格式如下：

列表名=［元素 1，元素 2，元素 3，元素 4，…］

列表是 Python 语言中使用最频繁的数据类型之一，列表中的每个元素都可以进行修改和删除。列表是有序的，每个元素的位置是确定的，可以用索引去访问每个元素。

Python 语言中，元素序号是从 0 开始的，如果输出列表中的第 4 个元素，那么对应的序号就是 3。如果想选取列表中的几个元素，可以使用切片操作符（：）来创建一个切片对象，格式为：列表名［序号 1：序号 2］。需要注意的是，其中序号 2 取不到，这是俗称的“左闭右开”，可采用只写 1 个序号的方法，取值取到最后 1 个元素。

［例 2-10］

```
X=['库存现金','银行存款','其他货币资金']  # 定义变量 X
Y=X[2]  # 调取第 3 个元素
print(Y)  # 输出第 3 个元素
```

运行结果如图 2-17 所示。

```
其他货币资金
```

图 2-17 ［例 2-10］运行结果

［例 2-11］

```
X=['库存现金','银行存款','其他货币资金','应收账款','应收票据']  # 定义变量 X
Y=X[1:4]  # 调取第 2～4 个元素
Z=X[1:]   # 调取第 2 个元素到最后
A=X[-3:]  # 调取倒数第 3 个元素到最后
B=X[:-2]  # 调取倒数第 2 个元素前的所有元素(左闭右开,不包含倒数第 2 个元素)
print(Y)  # 输出 Y
print(Z)  # 输出 Z
print(A)  # 输出 A
print(B)  # 输出 B
```

运行结果如图 2-18 所示。

```
['银行存款', '其他货币资金', '应收账款']
['银行存款', '其他货币资金', '应收账款', '应收票据']
['其他货币资金', '应收账款', '应收票据']
['库存现金', '银行存款', '其他货币资金']
```

图 2-18 ［例 2-11］运行结果

4. 元组

元组也是一种有序的数据结构，与列表类似，其中的元素可以是不同的数据类型，但元组一旦创建，其元素就不能修改。元组通过圆括号“()”表示，基本格式如下：

元组名=（元素 1，元素 2，元素 3，元素 4，…）

5. 字典

字典是一种映射类型，通过键值对存储数据，是无序的键值对（形式为 key：value）的集合。其中，键必须是不可改变的数据类型，如字符串、数字或元组，而不能是列表、字典、集合等可变数据类型。在同一字典内，键必须是互不相同的。字典通过大括号表示，基本格式如下：

字典名={键 1：值 1，键 2：值 2，键 3：值 3，…}

在字典中，每个元素都有两个部分，区别于列表中一个元素只有一个部分。前一个部分称为键，后一个部分称为值，中间用冒号相连，一个键对应一个值。

［例 2-12］

```
X={'库存现金':12000,'银行存款':500000,'其他货币资金':200000,'应收账款':180000,'应收票据':60000}    # 定义变量 X
for i in X:  # 循环字典中的元素
print(i+':'+str(X[i]))   # 输出键和值，键值中间用：相连
```

运行结果如图 2-19 所示。

```
库存现金:12000
银行存款:500000
其他货币资金:200000
应收账款:180000
应收票据:60000
```

图 2-19 ［例 2-12］运行结果

6. 集合

集合是一种无序的、不重复的元素集。集合通过大括号表示，但不支持通过索引访问元素。相对于列表和字典，元组和集合的使用频率不高。集合的基本形式如下：

```
集合名={元素 1，元素 2，元素 3，…}
```

2.3.3　程序控制流程

程序控制流程是程序设计中一个重要的内容。简单来说，流程就是计算机执行代码的顺序，流程控制就是对计算机执行代码的顺序进行有效的管理，通过流程控制才能实现程序中的业务逻辑。

在 Python 语言中流程控制分为以下三类。

顺序流程：这是 Python 语言最基本的程序控制流程，程序中的各个操作按照源代码中的排列顺序，自上而下依次执行。顺序流程中的操作是一步接一步，按顺序执行的，不需要关键字，也不需要特殊说明，直到所有语句都执行完毕。

选择流程：又称分支流程，允许程序根据某一步的条件判断，有选择地执行相应的代码块。常见的选择流程有：if…else、switch…case 等。例如，if…else 流程会根据条件的真假来决定执行哪一部分代码。

循环流程：在满足一定条件下，一直重复执行某段代码。常见的循环流程有 for 循环和 while 循环。循环流程可以细分为当型循环和直到型循环，前者先判断条件再执行，后者先执行再判断条件。

前面小节所编写的代码都是顺序流程，代码自上而下、一行一行地执行，本节主要学习流程控制中的选择流程和循环流程。

1. 选择流程

其实在工作、生活中，选择无处不在，人们每天都在进行各种各样的选择与判断，比如，如果下雨就不出门、只有年满 18 周岁才能去网吧等。同样地，在财务工作中，也常常需要进行条件判断，比如，只有业绩达标才能发放绩效奖金，个人所得税超过起征点才需要交税等。

选择流程根据选择的不同又可以分为单分支、双分支和多分支流程。在 Python 语言中常用 if 条件语句来控制分支流程。

1）单分支流程

单分支流程使用 if 语句即可，其语法格式如下：

```
if 条件：
    代码块 1（满足条件时要做的事情）
```

该语法表示如果条件为真，则执行代码块 1。条件语句后必须加上冒号，否则程序会报错。单分支流程图如图 2-20 所示。

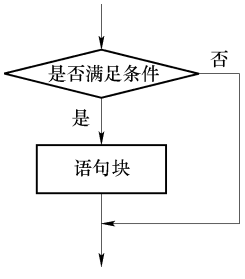

图 2-20　单分支流程图

［例 2-13］

如果分数达到 60 分，则及格

```
score=60  #分数为 60 分
if score>=60:   #如果分数大于等于 60 分
    print('恭喜您,及格啦!')  #输出'恭喜您,及格啦!'
```

运行结果如图 2-21 所示。

图 2-21 ［例 2-13］运行结果

2）双分支流程

单分支流程当条件满足时，才执行相应的代码，当条件不满足时，直接结束流程。但在实际工作中往往不止一个选择。如果是双分支流程，使用的是 if…else 语句，具体语法格式如下：

```
if 条件：
  代码块 1（满足条件时要做的事情）
else：
  代码块 2（不满足条件时要做的事情）
```

该语法表示如果条件为真，则执行代码块 1；否则，执行代码块 2。需要强调的是 else 语句后不包含条件。双分支流程图如图 2-22 所示。

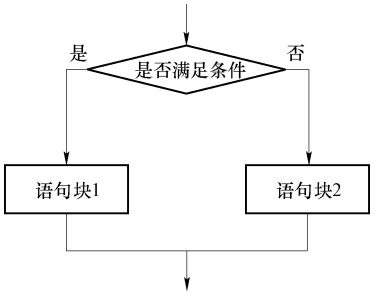

图 2-22　双分支流程图

［例 2-14］

输入分数，如果分数达到 60 分，则及格；否则，不及格

```
score=float(input('请输入您的分数:'))
if score>=60: #如果分数大于等于 60 分
    print('恭喜您,及格啦!')  # 输出'恭喜您,及格啦!'
else:
    print('很遗憾,您未及格!') #输出'很遗憾,您未及格!'
```

运行结果如图 2-23 所示。

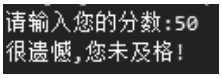

图 2-23 ［例 2-14］运行结果